CPS: Medium: GOALI: Enabling Scalable Real-Time Certification for AI-Oriented Safety-Critical Systems

Principal Investigator
James H. Anderson,* Kenan Distinguished Professor

Co-Principal Investigators
Frank D. Smith,* Research Professor
Ron Alterovitz,* Professor
Prakash Sarathy,† Chief Engineer/Senior Program Manager

*Department of Computer Science
The University of North Carolina at Chapel Hill
Chapel Hill, North Carolina 27599-3175
Phone: (919) 590-{6057, 6184, 6068}
E-mail: {anderson, smithfd, ron}@cs.unc.edu

†Mission Management Control Systems
Northrop Grumman Aerospace Systems
Redondo Beach California 90278
Phone: 310-812-6914
Email: sriprakash.sarathy@ngc.com

June 2020

Project Summary

Overview.

CPS: Medium: GOALI: Enabling Scalable Real-Time Certification for AI-Oriented Safety-Critical Systems,
James H. Anderson, PI, The University of North Carolina at Chapel Hill

In avionics, an evolution is underway to endow aircraft with "thinking" capabilities through the use of artificial-intelligence (AI) techniques. This evolution is being fueled by the availability of high-performance embedded hardware platforms, typically in the form of multicore machines augmented with accelerators. Unfortunately, avionics software certification processes, which are rooted in the twin concepts of *time and space partitioning*, have not kept pace with this evolution. On a uniprocessor, these concepts can be simply applied to decompose a system into smaller components that can be specified, implemented, and understood separately. On a multicore+accelerator platform, however, component isolation is much more difficult to achieve efficiently. This fact points to a looming dilemma: unless reasonable notions of component isolation can be provided in this context, *certifying AI-based avionics systems will likely be impractical*.

This project will address this dilemma through multi-faceted research in the CPS Core Research Areas of Real-Time Systems, Safety, Autonomy, and CPS System Architecture. It will contribute to Real-Time Systems and Safety by producing new infrastructure and analysis tools for component-based avionics applications that must pass real-time certification. It will contribute to Safety and Autonomy by targeting the design of autonomous aircraft that must exhibit certifiably safe and dependable behavior. It will contribute to CPS System Architecture by designing new methods for decomposing complex AI-oriented avionics workloads into components that are isolated in space and time.

Intellectual merit. This project will produce a framework for supporting components on multicore+accelerator platforms in AI-based avionics use cases. This framework will balance the need to isolate components in time and space with the need for efficient execution. Component provisioning hinges on execution time bounds for individual programs. New timing-analysis methods will be produced for obtaining these bounds at different safety levels. Research will also be conducted on performance/timeliness/accuracy tradeoffs that arise when refactoring *time-limited* AI computations for perception, planning, and control into components. Experimental evaluations of the proposed framework will be conducted using an autonomous aircraft simulator, commercial drones, and facilities at Northrop Grumman Corp.

Broader impacts. There has been a continuous push over the past 40 years toward more semi-autonomous and autonomous functions in avionics. This push began with auto-pilot functions and is increasingly being fueled by advances in AI software. Avionics certification procedures have not kept pace with these advances. This project will focus on a key aspect of certification: validating real-time correctness. The results that are produced will be made available to the world at large through open-source software. This software will include operating-system extensions for supporting components in an isolated way and mechanisms for forming components and assessing their timing correctness.

A special emphasis will be placed on outreach to girls and women. Such outreach will include: events involving the Graduate Women in CS (GWiCS) group at UNC, which hosts an annual research symposium targeted toward undergraduate women and other under-represented minorities; Tar Heel Hack, a hackathon for local middle and high school girls; UNC CS's Girls Who Code Club, which provides local girls in grades 6-12 with a community for learning about CS; and UNC CS's annual Open House and Science Expo. These events will include hackathon projects as well as interactive demos of autonomous systems.

Keywords: Real-Time Systems; Safety; Autonomy; CPS System Architecture

Project Description

1 Introduction

The use of artificial-intelligence (AI) techniques to endow embedded systems with "thinking" capabilities is transforming the role these systems are playing in our everyday lives. A decade ago, the notion of producing aircraft and automobiles at mass scales that can autonomously "think" may have seemed far fetched, yet we are closer than ever to this reality today. The evolution towards this new reality of highly intelligent systems may, in fact, be the most profound development in embedded computing to date.

Unfortunately, as this evolution moves forward, a fundamental stumbling block is looming: among the use cases that utilize the AI techniques fueling this evolution, some of the most compelling ones—such as autonomous aircraft and automobiles—fall within *safety-critical* domains for which *certification* is essential. The challenge such use cases present is that, in comparison to "traditonal" embedded systems, the AI-oriented workloads that must be supported are much more complex, and the hardware needed to support them—typically multicore machines augmented with accelerators—is much more complex as well. ***How should certification processes in safety-critical domains evolve to address these complexities? Can these processes be made to scale to large systems that may be subject to partial redesigns during their lifetime?***

Focus of this project. This project is directed at partially tackling these questions in the context of *avionics systems*. A complete certification solution would involve addressing a myriad of difficult verification-related issues, the full range of which is well beyond the scope of any single project. The specific focus of this project will be ***scalable real-time certification***. Specifically, we seek to devise techniques that enable a complex AI-oriented workload to be decomposed into manageable components that are isolated from one another, so that intra-component timing constraints can be verified independently. While our primary focus in this work will be on verifying timing constraints, we note that the ability to isolate components can aid in verifying logical correctness as well. It is our hope that the body of work produced in this project can be informative to the evolution of existing certification standards for avionics.

Avionics certification. While AI computations in automotive and avionics systems may share much in common, the latter have always been subject to much stricter certification. Fundamentally, existing avionics software certification processes are rooted in the ideas of time and space partitioning. These concepts allow a system to be decomposed into components that can be specified, implemented, and understood separately. Broadly speaking, *time partitioning* means that the real-time constraints of each system component can be certified independently, and *space partitioning* means that memory accesses by one component cannot adversely affect other components. On a conventional uniprocessor flight computer, these principles can be applied to fully isolate system components from one another, enabling a separation of concerns that is crucial for providing certification authorities with high confidence in a system's correctness.

Moving forward to AI-oriented avionics workloads and the hardware platforms they require, the situation becomes much murkier. For example, time partitioning now involves not only CPUs but accelerators as well. Also, for good performance, some degree of hardware sharing is inevitable, but any sharing breaks the illusion of isolation. Despite such complexities, ***avionics certification processes must evolve so that reasonable notions of isolation can be afforded to system components in these new use cases***.

Overview of proposed research. In this project, we seek to answer a single overarching question:

> ***Can a time and space partitioning solution be found for AI-oriented avionics workloads hosted on multicore+accelerator hardware that enables the real-time correctness of an overall system to be certified with high confidence while making efficient usage of hardware resources?***

We will address this question by pursuing five research goals.

- **Goal 1: Produce criticality-aware timing-analysis methods for multicore+accelerator platforms.** In order to support components, we need to know how long their constituent tasks take to execute, and these tasks may be of differing *criticalities*, with higher-criticality tasks requiring more stringent analysis. We will extend prior work on *measurement-based probabilistic timing analysis (MBPTA)* [16,21] to provide execution-time safety assurances based on criticalities in a multicore+accelerator setting.

- **Goal 2: Enable cross-component temporal and spatial isolation.** We will devise basic time- and space-partitioning mechanisms for multicore+acclerator platforms that approximate the uniprocessor ideal of giving components the illusion of executing on dedicated hardware. In producing such mechanisms, we will extend prior work on mitigating multicore interference (due to cache or memory contention, I/O, *etc.*) to apply in a component hierarchy where criticality-related nuances exist.
- **Goal 3: Create methods for validating component timing constraints.** Using the mechanisms developed under Goal 2, we will develop automatic methods for forming components and assigning hardware resources to them, and for validating intra-component timing constraints. AI computations are typically formulated as processing graphs, and the timing constraints of interest are graph response-time bounds. These graphs have complexities not considered in prior work.
- **Goal 4: Create methods for component-wise AI.** There has been a trend recently in research on AI for autonomy to aggregate functionality. Such aggregation can be harmful from a certification point of view (*and AI researchers need to be aware of this*). We will investigate different methods for decomposing avionics AI functions (*e.g.*, grouping by time horizon vs. by functionality) and assess the tradeoffs involved. We will also initiate work on *time-limited AI*.
- **Goal 5: Evaluate our isolation methods in supporting "real" AI-oriented avionics workloads.** We will conduct evaluations through experiments conducted at UNC using both Microsoft AirSim [82] and sub-scale drones. Additional evaluations will be conducted by our students using design and testing facilities for autonomous aircraft at Northrop Grumman as part of summer internships.

Qualifications. PI Anderson has published extensively on the topic of multicore real-time systems and has taken a leading role internationally in work in this area. He is an ACM Fellow and an IEEE Fellow, and his fellow citations reference this research track record. Co-PI Smith has several decades' worth of software-development experience (both at IBM and at UNC) in the areas of operating systems (OSs), compilers, networking, and file systems (*e.g.*, as a principal in the Andrew Project). Co-PI Alterovitz brings expertise on AI and robotics to the project. In 2019, he received the Presidential Early Career Award for Scientists and Engineers (PECASE). This is the highest honor bestowed by the US government to early-career scientists and engineers. Co-PI Sarathy is the mission-management technical lead at Northrop Grumman Corp. for several DoD projects directed at advanced avionics for unmanned systems. His areas of technical expertise include avionics applications of neural networks in decision-making paradigms.

Organization. In the rest of this proposal, we provide needed background (Sec. 2), describe the research we propose to undertake and our experimental and collaboration plans (Secs. 3-5), discuss broader impacts (Sec. 6), and discuss results by the investigators from prior NSF support (Sec. 7).

2 Key Concepts and Related Work

In this section, we present needed background on trends in avionics, the use of AI techniques in realizing autonomous functionality, avionics certification, and real-time systems. Due to space constraints, *our overview of prior work focuses on research of direct relevance to our research agenda*.

2.1 Emerging Trends in Avionics

Smallish drones (which usually do not require certification) have seen greater innovation in AI-based autonomy than larger aircraft (which do), yet autonomy in the latter has been increasing for 40 years.

Evolution of autonomy in avionics. The most successful instance of autonomy has been the *autopilot* function installed in piloted fly-by-wire aircraft in the mid-1980s by Boeing and Airbus [91]. Autopilot initially provided flight control for cruising, but more advanced functions are now integrated into a full *flight-management system (FMS)* that enables autonomous control from just after take-off to landing [83]. Conventional FMSs are firmly rooted in control theory, with control laws implemented as ordinary sequential programs without the learning or adaptation capabilities inherent in AI-based applications.

Autonomy for unmanned aerial vehicles (UAVs). A UAV is essentially a "remotely piloted aircraft." Notable (large) military UAVs include the MQ-9 Reaper (offensive strikes), the RQ-4 Global Hawk (surveillance and reconnaissance), and the MQ-1B Predator (multi-mission). In these systems, the ground-based crew is responsible for mission control and strikes, but flight operation is mostly autonomous, using control laws similar to those in conventional FMSs. More recently, UAVs providing even greater autonomy have been demonstrated, such as the Navy's MQ-25, an unmanned aerial refueling tanker.

In the private sector, smallish UAVs—referred to as *drones* in this document—weighing up to 55 pounds and costing $1,000–$10,000, have been licensed for operation by the US Federal Aviation Administration (FAA). As of March 2020, over 440,000 commercial drones had been registered with the FAA [25]. Camera-equipped drones are used in applications in fields as varied as agriculture, construction, real estate, and law enforcement, to name a few.

Hardware trends. In large aircraft, the dominant hardware infrastructure is based on the concept of *integrated modular avionics (IMA)*, which is defined in several ARINC (Aeronautical Radio, Inc.) reports (the core ones being 629, 651, and 659) [74]. The goal of IMA is to integrate all major aircraft systems (*e.g.*, flight management) onto a few (ideally one) hardware modules. The canonical module is a cabinet with a power supply, a bus, and plug-in slots for processors, memory, sensors, and I/O, as shown in Fig. 1. In contemporary IMA systems, a slot may hold a plug-in board containing a system-on-chip (SoC) with a multicore processor and integrated GPU along with on-board memory. Configurations with separate slots for a multicore processor, a discrete GPU or other accelarator, and shared DRAM are also possible if applicable ARINC specifications are met. More recently, embedded systems meeting *size, weight, power, and cost (SWaP-C)* requirements are providing much greater computing power with SoC platforms that include accelerators.

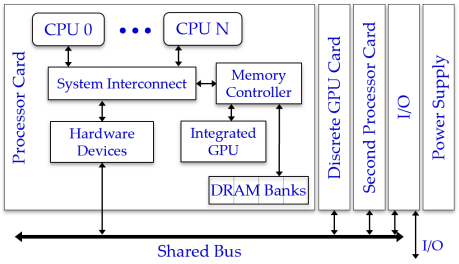

Figure 1: A sample hardware platform.

Software trends. The IMA concept is rooted in the sharing of hardware among software components, which can create non-deterministic effects due to interference. The IMA design recognizes this issue and includes a real-time OS (RTOS) design specification in ARINC report 653 [75]. The fundamental OS concept in ARINC 653 is a *partition*. A partition encapsulates a set of related software components and affords them isolation in *time* (execution) and *space* (memory). Space isolation is provided via functions for memory protection. Time isolation is achieved with a two-level scheduling mechanism. Time slices are first allocated to partitions, and, within each partition, an in-partition scheduler allocates time to its contained tasks. On multicore platforms, the current specification allows only one partition per core[1] (effectively obviating partition time slicing) but a single partition can execute on multiple cores, as shown in Fig. 2.

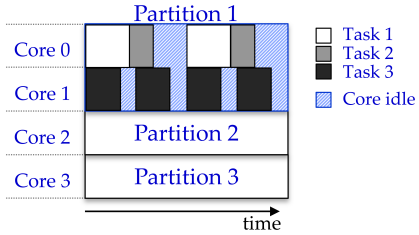

Figure 2: Three partitions on four cores. Partitions 2 and 3 each execute on one core, while Partition 1 executes on two cores. In-partition scheduling is shown for Partition 1.

Due to the extreme limitations on SWaP-C, avionics software for autonomous functions in drones has been mostly limited to basic flight controls and GPS navigation. As a result of the increasing computational capability of embedded hardware platforms, however, computationally demanding AI functions are being used on several commercial and small military drones, such as the RQ-11 Raven and Desert Hawk III.

2.2 AI Applications and Workloads for Autonomy

In this section, we review two highly promising uses of AI in avionics for autonomous operation.

Computer vision (CV). One of the most fundamental capabilities an autonomous aircraft must have is *perception*, *i.e.*, the ability to sense and understand its environment. Cameras arguably provide the richest and most cost-effective sensing capabilities. With one or more cameras operating at rates fast enough to convey motion information, CV algorithms can effectively provide the required perception capabilities.

[1]This restriction is under study for change. The results of this project would be highly informative in making this change.

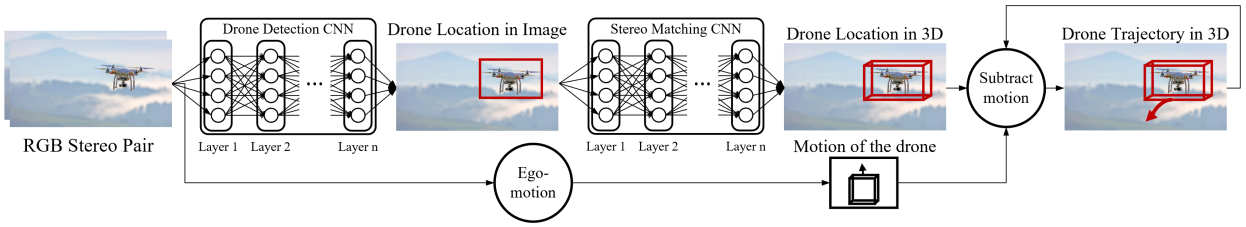

Figure 3: Example drone detection/tracking graph, comprised of two deep neural networks and additional nodes.

Example. Fig. 3 illustrates how CV algorithms might be used to detect and track other drones that may cross a given drone's flight path. First, the drone performs *object detection* to identify any other drones. This is done using a *convolutional neural network (CNN)* to quickly obtain a *bounding box* for any drone. Second, the drone must determine whether another drone is a collision threat. This is done by analyzing the other drone's relative position in 3D via another CNN performing *stereo matching* [50], and then estimating the other drone's motion relative to *ego-motion* (*i.e.*, self-motion). Next, *object tracking* is performed by remembering and tracking any detected drone's position over time. CV methods can then build a *3D geometry map* of the environment indicating the location and motion of obstacles, such as other drones.

Planning and control. Another important capability of an autonomous aircraft is for it to automatically decide where to move, including both short-term decisions (to react to obstacles such as other drones) and longer-term decisions to accomplish its mission. Planning algorithms typically compute waypoints for the aircraft to reach that accomplish the aircraft's task while avoiding obstacles represented in the 3D geometry map, and control algorithms enable the drone to move through the waypoints. Modern control methods for autonomous drones often use machine learning, *e.g.*, evaluating a neural network or other learned model that was pre-trained on a combination of simulation and physical drones to enable fast motion in highly dynamic settings that are challenging to model using traditional control approaches [37].

Importance of accelerators and isolation. The neural networks used to realize these two capabilities consist of a number of sequentially executed "layers," many of which run convolutions with matrix-matrix operations [50]. Convolutions are computationally expensive when implemented by a single CPU thread, but can be structured to run on hundreds of parallel threads/cores on a GPU or some other accelerator.

Advances in multicore+GPU embedded platforms are already enabling AI capabilities in commercial drones. For example, Skydio 2 can autonomously avoid collisions while following a tracked object by using an NVIDIA Jetson TX2 with six ARM cores and a 256-core GPU to process six camera feeds [84]. Enhanced autonomy will require supporting many more accelerator-using computations, which will be difficult to certify if not isolated from one another. Unfortunately, ***accelerator sharing breaks isolation.***

2.3 Avionics Software Certification

In the US, the FAA is responsible for certifying the safety and airworthiness of commercial aircraft.[2] The European Union Aviation Safety Agency (EASA) performs the same function in the EU (using similar processes as the FAA). The FAA process involves a byzantine network of related (and often overlapping) documents for regulations, advisories, directives, orders, notices, *etc.*, as well as a collection of documents providing "guidelines," "considerations," and "positions," which are actually de facto standards [24].

Software certification. For the certification of avionics software (this project), the primary documents are ARP 4761 [86] for the *safety-assessment process*, ARP 4754 [78] for the *development process*, and DO-178C [76] for the *compliance process*. DO-178C is supplemented by documents that address specific tools and technologies recommended: DO-248 (clarifications and rationales), DO-330 (tool qualification), DO-331 (model-based development), DO-332 (object-oriented technology), and DO-333 (formal methods).

ARPs 4761 and 4754 define five *design assurance levels (DALs)*, often called *criticality levels*, based on the severity of software-failure impacts. As shown in Table 1, DO-178C refers to the same five levels and defines verifiable per-level objectives. High-level *requirements* are defined for software components according to their designated criticality level. These requirements are then refined into detailed requirements for the processes, modules, function calls, and program statements in the component.

[2]The US military has its own airworthiness certification agencies but has the option of FAA certification.

Level	Failure Condition	Interpretation
A	Catastrophic	Failure may cause a crash.
B	Hazardous	Failure has a large negative impact on safety or performance ...
C	Major	Failure is significant, but has a lesser impact than a Hazardous failure ...
D	Minor	Failure is noticeable, but has a lesser impact than a Major failure ...
E	No Effect	Failure has no impact on safety, aircraft operation, or crew workload.

Table 1: DO-178C criticality levels.

We are mainly interested in *real-time requirements* like "task τ_i must issue a cockpit warning within 100ms of stall detection." Ensuring such requirements requires knowing task *worst-case execution times (WCETs)*. FAA document CAST-32A [23] gives guidance for determining WCETs on multicore platforms.

Regarding accelerators, the FAA has approved the use of GPUs to realize display functions. However, in this project, we are interested in using accelerators to speed critical planning and control steps.

2.4 Real-Time Systems

This project is aimed at avionics systems comprised of *components*, some providing AI functions, that must be isolated from one another. In reality, such a system would likely transition among different *functional modes* in response to external events, with only certain components (or portions thereof) being active in a given mode. For conciseness, we only consider the problem of ensuring isolation in a single mode herein, though all of the work we discuss can be generalized to multi-mode systems.

As we have seen, AI computations tend to be expressed as processing graphs. While AI graphs often have cycles, the most widely studied notion of a *real-time* processing graph is that given by the *sporadic DAG (directed acyclic graph) model* [13]. Under this model, graph nodes represent tasks (programs), and edges denote precedence constraints. For example, in Fig. 4, the i^{th} invocation, or *job*, of task τ_2 cannot start execution until the i^{th} job of task τ_1 completes. A sporadic DAG G has a designated *source* node with no incoming edges. The entire graph G is implicitly executed cyclically, with successive invocations of its source being at least T_G time units apart, where T_G is G's *period*. The main timing constraint of interest is an *end-to-end response-time bound*, which indicates the maximum allowed time for a given graph invocation to complete.

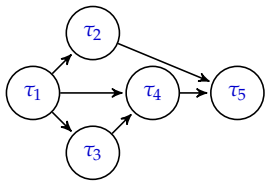

Figure 4: A DAG example.

We hereafter assume that each component of interest is comprised of a set of processing graphs, but the graphs we consider generalize sporadic DAGs in several ways. First, a graph may have multiple sources (*e.g.*, to support graphs that fuse data from different sources that execute at different rates). Second, each graph node is assigned a *criticality level* (as in DO-178C). Third, graphs may have cycles (*e.g.*, to enable processing historical data, as in Fig. 3).[3] Fourth, graph nodes may access accelerators. Fifth, certain graph nodes may be allowed to execute in parallel to lower response times. Note that the computational model assumed here is quite general—even an ordinary task can be viewed as a (single-node) graph.

Timing analysis vs. schedulability analysis. The goal of *timing analysis* is to produce an execution time bound C_i for each task τ_i. Usually, a single WCET is required, though our mixed-criticality context is more nuanced (see below). The goal of *schedulability analysis* is to verify that timing constraints are met (in our case, that end-to-end response-time bounds are acceptable). Of relevance to schedulability is the processor share required by each task τ_i, as given by its *utilization* $U_i = C_i/T_G$. When assessing schedulability on a multiprocessor, it is often necessary to restrict utilizations in some way. This results in *capacity loss*, *i.e.*, unavailable processing capacity. In addition to such *schedulability-induced* capacity loss, *overhead-related* loss can occur due to OS activities, task migration costs, *etc.* *Interference* across cores on a multicore platform can also cause capacity loss. ***The rigid ARINC 653 scheduling strategy for partitions is subject to severe capacity loss***, mainly because it allocates cores at the granularity of whole partitions.

Mixed-criticality analysis. Pessimistic WCETs can induce significant capacity loss. A technique for mitigating this loss was proposed by Vestal, who (while working in the avionics industry) proposed dealing

[3] If tasks are sequential, then each node implicitly has a self-loop. Here we are referring to more complicated cycles.

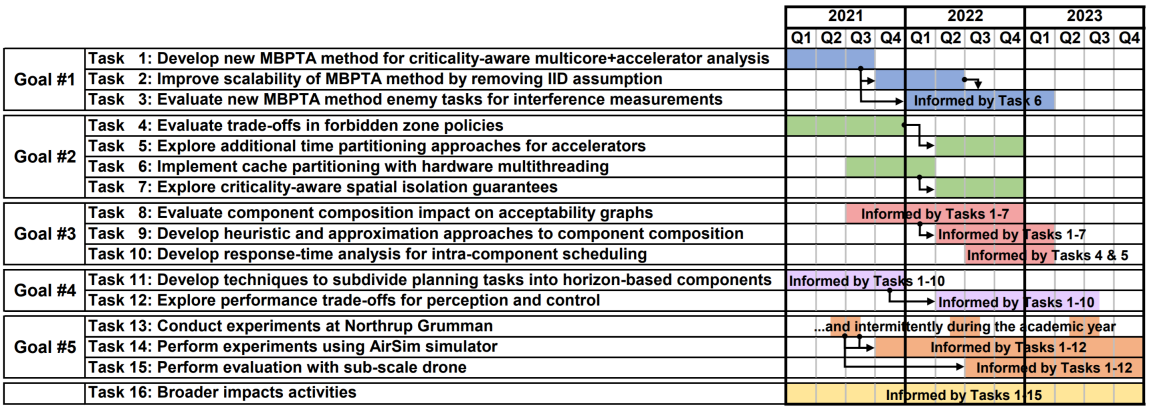

Figure 5: Gantt chart of the project timeline. Intra-goal task dependencies are shown with arrows, and inter-goal dependencies are labeled with text.

with less-critical tasks less pessimistically [94]. Under his proposal, each task in a system with L criticality levels has a *provisioned execution time (PET)* specified at every level, and L system variants are analyzed: in the Level-ℓ variant, the schedulability of all Level-ℓ tasks is verified with Level-ℓ PETs assumed for all tasks (at any level). The degree of pessimism in determining PETs is level-dependent, with more conservatism applied to higher levels. We intend to apply similar ideas in the context of processing graphs.

2.5 The Looming Certification Dilemma

After many years of trying, industry and the FAA/EASA seem to be finally converging on a basic approach to safety certification for avionics software on multicore platforms. However, this basic approach, while rooted in time and space partitioning, may result in significant capacity loss by overly restricting the allocation of cores to partitions. ***Such loss will limit the range of future applications that can be supported.***

Perhaps of even greater concern is a looming set of issues beyond multicore. New hardware in the form of *accelerators (especially GPUs)* will soon be required for emerging AI-oriented avionics. Current certification methods and tools cannot deal with accelerators. Consider that a CV application written as a sequential program actually executes partially on a CPU core and partially on a GPU, and these executions may overlap in time. The GPU execution is usually not preemptable while the CPU is. Further, efficient use of a GPU may require sharing it among different components. These factors greatly complicate all aspects of certification. ***Multicore CPUs and accelarators must be treated holistically for certification.***

3 Research Description

In this project, we will resolve the avionics certification dilemma described above by pursuing a research agenda that focuses on five goals. First, we will strengthen the certification process by producing criticality-driven timing analysis that targets multicore+accelerator platforms. Second, we will produce methods for isolating different system components in space and time on such platforms while allowing some degree of hardware sharing. Third, based on these methods, we will devise techniques for creating components and validating their timing constraints. Fourth, we will elevate research on AI for autonomy by making time and component-wise certification first-class concepts. Fifth, we will experimentally assess the efficacy of the methods we produce using an advanced avionics simulator, sub-scale drones, and facilities at Northrop Grumman. We elaborate on the first four of these goals below and consider the fifth in Sec. 4. Our research agenda maps to a three-year research plan, which is depicted in Fig. 5.

Hardware platform. To facilitate discussing our research goals, we first describe our intended hardware platform, the AMD Ryzen 9 3950X, which is illustrated in Fig. 6. This platform has 16 physical cores (shown in blue), each supporting two simultaneous hardware threads, subdivided among four clusters (shown in grey) with a shared L3 cache each—a design intentionally similar to ARM's next-generation embedded CPU [92]. The 3950X supports AMD's Quality of Service extensions [6], which include hardware support for partitioning L3 caches and memory buses (shown in green). The 950X contains 24 PCIe

4.0 lanes, providing 48 GB/s of bandwidth for accelerators. With this high bandwidth, up to 12 GPUs could be connected.

In our initial work on hardware accelerators, we intend to use AMD Radeon VII GPUs (shown in red). The Radeon VII is one of AMD's most powerful GPUs. Our choice of AMD over the market leader NVIDIA is due to the fact that AMD GPUs offer *an open-source software stack* (unlike NVIDIA) and *hardware partitioning support* [70]. ***Openness is essential from a safety point of view***.[4] As the project advances, we will likely consider other accelerators (both GPUs and FPGAs), and more of them.

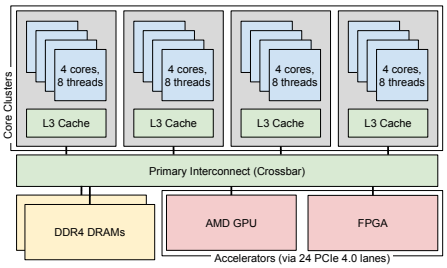

Figure 6: AMD Ryzen 9 3950X.

Software platform. We will implement the isolation mechanisms developed in this project in LITMUS^{RT} [1], a real-time variant of Linux developed by our group. Our choice of LITMUS^{RT} over a commercial RTOS is largely due to its open-source nature and our familiarity with it. Also, the point of this project is to demonstrate the efficacy of resource-allocation *principles* that can be applied in a variety of RTOSs.

3.1 Goal 1: Produce Criticality-Aware Multicore+Accelerator Timing Analysis

Supporting isolated components on a multicore+accelerator platform entails adequately provisioning hardware resources for components, but to do so, we need valid task execution costs. This observation gives rise to our first research challenge: *Given a task with a specified criticality level and a (possibly empty) set of dedicated hardware resources, how can we determine execution costs that account for contention on non-specified hardware resources for all relevant criticality levels?* Such a task may execute on a CPU and/or accelerator.

Problems with existing timing analysis. Existing timing-analysis methods fall short in this setting. The classical approach is *static timing analysis*, which relies on detailed knowledge of the task being executed and the platform executing it. When multiple tasks/components can interfere with one another, knowing the exact state of the platform becomes essentially impossible, preventing static analysis. This problem is even worse when dealing with accelerators that have black-box components, such as many GPUs.

An alternative to static analysis is *measurement-based probabilistic timing analysis (MBPTA)*, which attempts to probabilistically bound task execution times by examining a relatively small sample of jobs [16, 21]. The majority of recent MBPTA work has focused on a family of methods known as *extreme value theory (EVT)*. Traditionally, sound EVT analysis has required observed execution times to be identically and independently distributed (IID) [11, 27, 72], but such times will often have statistical dependencies. There is currently no agreement on how to work around this issue [16, 21, 49]. Both a review of the literature on EVT and our own preliminary experiments show it is not useful for timing analysis in a safety-critical context. But if neither static analysis nor EVT are viable options, how can we build a certifiable system?

A new MBPTA method. We propose to develop a new version of MBPTA that acknowledges that the measurements used to build any estimates are themselves a source of uncertainty and subject to random variation. In particular, we will build our schedulability analysis around *safe WCETs*, or sWCETs, where each task τ_i has an sWCET S_i^q, determined as a function of a sample of execution times, such that when E_i is a random variable corresponding to the execution time of a randomly selected job of τ_i, $\Pr(E_i \leq S_i^q) = q$ holds; the ideal situation is to have q approach one. In this definition, *both* E_i and S_i^q are random variables; consequently, the value of q accounts for not only variations in E_i, but also the possibility that the execution time samples used to determine S_i^q does not give a good idea of the true distribution of τ_i's possible execution times. Existing MBPTA methods do not generally account for the latter factor. When a single task needs costs at multiple criticality levels, higher q values will correspond to higher criticality levels.

In recent work [68], we provided one possible method for determining S_i^q for a given value of q, or, alternatively, the appropriate value of q for a given S_i^q value. This method relies on the assumption that our observed execution-time samples constitute IID samples from τ_i's true execution-time profile, but we found empirically that our method produces accurate results even when samples are not IID.

[4] Clearly, safety-critical designs will exist with proprietary details, but such a design cannot be truly safe if it is based on an underlying substrate with unknowable behaviors. Royalty-free, open APIs are actually key to industry adoption.

In this project, we plan to expand on this work by improving scalability—in our current method, q increases slowly with respect to the number of samples used—and reducing reliance on the assumption that samples are IID. One promising way forward may be using the Saw variant of Chebyshev's Inequality [80], which estimates population extremes based only on sample mean and variance and does not rely on data being IID. In addition, we will develop a method to determine q values that are appropriate for specific criticality levels as specified in the FAA System Safety Handbook [26]. Since we need our execution times to be valid when hardware resources are accounted for, our sample measurements need to be obtained when tasks are competing for resources against "enemy" tasks designed to create high levels of interference. We will develop enemy tasks for this purpose starting from the techniques recently described in [43]. A similar approach should work for analyzing contention on accelerators. One advantage our method will have is that, since we do not require any platform knowledge, but only observed measurements, we are well-placed to develop sound timing analysis should the need arise for some black-box accelerators.

3.2 Goal 2: Enable Time and Space Partitioning on Multicore+Accelerator Platforms

Our work under Goal 2 will be directed at developing the requisite *mechanisms* that can be applied to isolate components in space and time. The *effective use* of such mechanisms is the focus of Goals 3 and 4.

We seek to allocate both processing (CPU and accelerator time) and non-processing (caches, memory, buses, *etc.*) resources required for exclusive use by a given component $\mathcal{P}$. We assume that $\mathcal{P}$ consists of one or more processing graphs (see Sec. 2.4) and has a specified *reservation* indicating its needed hardware resources. Working out the exact details of an effective notion of a "reservation" will be a research issue we need to resolve. For our purposes here, it suffices to consider a notion of a reservation defined by three parameters, C, T, and H, indicating that $\mathcal{P}$ is active for a duration of C time units during each period of T time units, and when active, it requires exclusive access to a set H of hardware resources; H would, for example, specify CPU and accelerator requirements as well as cache and memory partitions. All hardware resources used by $\mathcal{P}$ that are not specified in H would be considered as contended by other components.

Time partitioning. Any time-partitioning strategy must ensure that $\mathcal{P}$ has exclusive access to all processing resources in H when it is active. For the sake of concreteness, we propose one such strategy here, though others will be investigated in this project. In the specific approach here, we ensure the time partitioning of CPU resources via budgeting: during each period of time T, $\mathcal{P}$ is allocated a *time slice* of length C, and during this time slice, it has exclusive access of all CPUs specified in H. Dealing with accelerators is a bit trickier. For them, we propose to employ the concept of a *forbidden zone* [35], as depicted in Fig. 7. To ensure that accelerator usage completes before a time-slice boundary, accesses requested too near the end of a time slice (within the forbidden zone) may be delayed until $\mathcal{P}$'s next time slice.

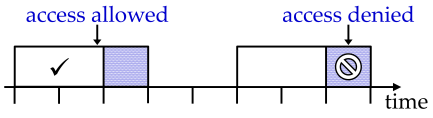

Figure 7: A component $\mathcal{P}$ with $C = 3$ time units, $T = 5$ time units, and a 1-time-unit forbidden zone.

We propose two forbidden-zone variants. The first is pessimistic: all accelerator accesses within a forbidden zone are prevented. The second is more optimistic but requires the ability to preempt accelerator use: access is allowed within the forbidden zone, but execution will be aborted if the time slice ends (the access must be re-tried in the next time slice). We will explore many trade-offs here. For example, short accelerator accesses may be allowed to optimistically execute, but long accesses may be prevented. Also, we will explore forbidden zones of different durations (*e.g.*, a common per-slice duration vs. durations that are per-resource or per-resource request) as well as dividing zones into optimistic and pessimistic regions.

Space partitioning. On a multicore+accelerator platform, space partitioning involves more than simply ensuring memory protection and policing the flow of information across components. In particular, the illusion of space partitioning can be broken on such a platform when concurrently executing components access common non-processing resources such as caches, memory, buses, *etc.* In recent years, the issue of eliminating interference with respect to such resources has been investigated extensively by the real-time systems community [3–5, 10, 18, 19, 29, 31–33, 45, 53–56, 59–61, 71, 89, 96, 97, 101, 102], including by our group [9, 17–19, 19, 56–59, 96]. However, in the context of this project, several new issues arise that need to be resolved. We mention two such issues here to provide a sense of the scope of this work.

The first issue pertains to the use of hardware multi-threading, which the FAA has flagged as being of importance to enable in avionics use cases [66]. Our 3950X hardware platform enables higher performance by providing two hardware threads per core. Unfortunately, concurrent thread execution on the same core can create unacceptably high interference for highly critical tasks. In contrast, for less-critical tasks, hardware threading may be very useful. We plan to investigate this issue, and examine the feasibility of partitioning both the L2 and L3 caches on the 3950X (through a combination of page coloring [51] and the 3950X's Quality of Service extensions) on a per-thread basis when threading is in use. Note that enabling hardware threading has implications for the timing-analysis research proposed under Goal 1.

The second issue arises due to our intent to elevate these isolation issues from the level of mere tasks to *components*. In a holistic sense, we envision the handling of non-processing resources as being *hierarchical* in our context. Across components, we seek to maintain strong isolation invariants, but within components, hardware-isolation techniques also may be of value, but in a less-strict way that hinges on criticalities. For instance, within a component, less-critical tasks may be provided with weaker isolation guarantees than highly critical tasks Also, within a component, it might be reasonable to partition a GPU into multiple *virtual GPUs* [44], but across components, we want no concurrent sharing of accelerators whatsoever.

3.3 Goal 3: Create Methods for Validating Component Timing Constraints

Goal 3 is directed at the problem of determining a (C, T, H) reservation for each component $\mathcal{P}$ such that cross-component interference is minimized (using the mechanisms from Goal 2) and all timing constraints are satisfied, assuming known per-criticality-level task execution times (obtained using the methods from Goal 1), each a function of the allocated hardware resources.

Abstracting processing resources as an exclusive virtual platform. A component $\mathcal{P}$ with a reservation (C, T, H) has exclusive access to each processing resource (CPU or accelerator) specified in H for a fraction of time given by C/T. In describing some of the research proposed below, it is convenient to view $\mathcal{P}$ as if it executes on a dedicated virtual platform consisting of possibly fewer CPUs and accelerators (as specified by H) than the actual platform, each executing at a slower speed, approximately C/T.[5]

Acceptability graphs. Given this notion of a virtual platform, we can reflect the space of viable reservation choices for a given component $\mathcal{P}$ by means of an *acceptability graph* whose axes specify the number of allocated processing resources and their speeds; a 2D version of such a graph is shown in Fig. 8. In such a graph, we define the set of all points such that the corresponding reservation is sufficient to ensure $\mathcal{P}$'s timing constraints as $\mathcal{P}$'s *acceptable area*. This area gives the reservation choices that are viable for $\mathcal{P}$.

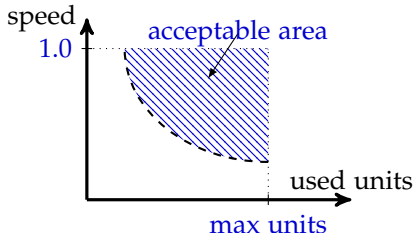

Figure 8: An acceptability graph.

Exploring reservation options. To understand how to best provision systems, we need to explore various issues concerning the pros and cons of different reservation choices. We mention two such issues here.

The first issue involves *component composition*. As the number of accelerators is limited, it might be necessary to combine multiple components into one, *e.g.*, it would not be reasonable to allow one component that uses all accelerators on our 3950X platform to execute alone if it requires only (say) four of the 3950X's 16 cores. To decide which components can be combined, we need to understand how the acceptability graphs for two different components $\mathcal{P}$ and $\mathcal{P}'$ *compose—i.e.*, what does the graph for $\mathcal{P} \cup \mathcal{P}'$ look like?

The second issues involves *over-provisioning*. One way to attempt to avoid having to re-provision a system as components are modified during its service lifetime is by giving its components more hardware resources than they minimally require. This observation gives rise to several questions. What factors would favor one over-provisioning choice over another? How do we determine which hardware resources are the real "pinch points" that might require re-allocation if not over-allocated? If a component is over-allocated resources, then spare resource capacity will be available at runtime—how should it be utilized?

[5]This notion of a virtual platform suffices to highlight some of our research ideas but over-simplifies matters. For example, due to limits on or contention for non-processing resources (*e.g.*, cache and memory space), the speed could be lower. Unfortunately, we lack sufficient space to properly delve into these and other nuances.

Allocating components. Given solutions to the problems described above, we will develop methods that determine which components need to be combined, the (C, T, H) reservation parameters for all resulting components, and a corresponding periodic timeline schedule. Our problem is related to bin-packing in that we can view cores as "bins" to be allocated but has many additional nuances. As bin-packing itself is NP-complete in the strong sense, we will likely need to resort to finding heuristics, though exact solutions might be possible that are reasonable for small problem sizes.

Scheduling. Given a component $\mathcal{P}$ with reservation (C, T, H), we need to determine how to schedule its contained graphs and compute response-time bounds for them. While real-time graph scheduling has been studied extensively before—representative publications include [7,12,22,46–48,62,81,98,99]—several extensions to prior work will be required for our purposes. Most importantly, scheduling in our context is within a "limited processing supply" setting, where the component $\mathcal{P}$ is active for only C time units out of every T time units. Also, the graphs that are our focus entail interacting with accelerators, can have multiple sources with different rates, and can have nodes of different criticalities. Finally, these graphs may contain cycles (as in Fig. 3), *i.e.*, they are not necessarily DAGs. While some of these issues have been considered to a limited extent before (*e.g.*, obviating multiple source rates by converting to a large single-rate DAG [28,79,88,93]), no response-time analysis exists that fully addresses them all.

3.4 Goal 4: Create Methods for Component-Wise AI

In Goal 4, we will refactor the AI components of an autonomous drone into components amenable to real-time certification. We will take into account that the effectiveness of a drone's AI components in real-world systems will ultimately depend on both the quality of the AI algorithms' results (*e.g.*, the accuracy of the CV algorithm or the ability of a planner to compute an obstacle-avoiding path) as well as the timing of those results, since a perfect solution computed too late may be useless.

AI software of autonomous drones. The AI software for an autonomous drone typically includes three major aspects: perception, planning, and control. *Perception* involves reading raw sensor data and translating it into a representation that is useful to planning and control. Common low-cost sensors on drones include GPS, inertial measurement unit (IMU), and cameras, and the perception component processes data from these sensors using CV methods to estimate the drone's location and to determine the presence, type, and movement of obstacles in its environment, as shown in Fig. 3. *Planning*, sometimes referred to as *navigation*, involves determining paths for the drone so it moves to accomplish an objective in a safe manner that avoids obstacles. *Control* involves computing low-level commands (*e.g.*, electric currents for each of the drone's motors) such that the drone moves along the path computed by the planner. Due to space constraints, we will focus our discussion below on planning and control. Due to the need to avoid obstacles, planning and control are particularly relevant for autonomous drones operating in cluttered environments, *e.g.*, indoors and in low-altitude outdoor flying in urban environments.

The need for time-limited AI. To improve the dynamic performance of autonomous vehicles, recent research approaches are increasingly integrating perception, planning, and control, often using machine learning to achieve the integration (*e.g.*, [30,63,77,85,87]). For example, an integrated control and planning component computes both a path and an associated feedback controller for that path. This tight integration can enable a drone to avoid dynamic obstacles more effectively and at a higher speed by considering the full range of available motor torques rather than a simplified motion model when computing a path to avoid obstacles. Also, perception can be blended with both planning and control by learning mappings directly from pixels to motor torques. However, a growing danger of tightly integrating perception, planning, and control is that an autonomous drone's AI increasingly becomes one large monolithic component. A large component has the potential to increase the quality of the AI's results, but also makes it increasingly challenging to ensure that the computation completes in a short, bounded time period. We believe time limits must be considered as a fundamental constraint when designing AI software for drones. For an AI algorithm to be effective for an autonomous drone, it must **(i)** be of high quality, *e.g.*, a CV algorithm accurately detects obstacles and a planner effectively avoids them, and **(ii)** satisfy timing constraints, *e.g.*, it is not useful to accurately detect an obstacle only after it can no longer be avoided. Hence, ***the effectiveness of an AI algorithm depends on <u>both</u> the quality and the timing of the result***.

Creating horizon-based certifiable components. We will investigate approaches to refactor the AI of an autonomous drone into time-limited AI components that will facilitate real-time certification. One approach we will consider is creating AI components based on time horizons, *i.e.*, the amount of time into the future that the component considers when generating its results. For example, a short-term planning and control component with a 1-sec. horizon is useful for avoiding collisions with high-speed obstacles since it allows the drone to consider nearby objects in its environment that it might encounter in the next second. A medium-term component with a horizon on the order of 10 secs. would enable efficient motion in cluttered environments where many obstacles may be present. A long-term component with a horizon of minutes or hours would enable planning for mission goals, *e.g.*, waypoints that enable the drone to reach a far away destination or to inspect a region of interest. We will consider components for each horizon that are amenable to determination of safe WCETs and acceptable response-time bounds, which should be substantially shorter than the component's time horizon. Specifically, we will investigate components for these time horizons that blend planning and control, including model-predictive controllers, learned controllers, and search-based geometric planners [20, 85, 87]. To reduce duplicative computations, we will also investigate approaches to enable the planning and control components with different horizons to share some information without too adversely impacting time and space isolation guarantees.

3.5 CPS Research Focus

This project will break new ground on many fronts in the core CPS research areas of ***real-time systems***, ***safety***, ***autonomy***, and ***CPS system architecture***. Notably, it will show how to decompose complex avionics software that runs on complex hardware into manageable components that are separated in space and time, while allowing for efficient platform usage. The need to evolve avionics software systems in this way has been widely recognized in industry, but avionics companies need the help of experts on real-time resource-allocation tradeoffs such as ourselves to make this evolution happen. By thoroughly delving into these tradeoffs, this project can be a key enabler for future avionics certification procedures that can accommodate forward-looking AI-oriented features. ***The CPS research community can produce paper after paper about the importance of these features, but they will never become part of certified avionics systems unless these certification procedures evolve!*** In experimental efforts (see Sec. 4), we will explore the safe integration of the overall component-based framework arising from our work within the control harness of advanced avionics simulators, sub-scale avionics systems, and systems at Northrop Grumman. We will also provide an open-source implementation of this framework in LITMUS^{RT} to provide a straightforward path for integration into commercial RTOSs. While the focus of this project is avionics, our results will be of use in any CPS domain of a safety-critical nature where complex software must be hosted on complex hardware, and a decomposition of a system into components would be useful for certification—notable domains of this kind include autonomous automobiles, medical robots, space vehicles, *etc.*

4 Evaluation/Experimentation Plan

This section describes our *integration* of the "cyber" results achieved by Goals 1-4 with "physical" drone hardware, and our intended evaluations.

Goal 5: Evaluate our Methods in Supporting "Real" AI-Oriented Avionics Workloads

This proposal claims that our results will enable the development of autonomous functions for avionics where *temporal safety* is the critical requirement. Our evaluation experiments will use both simulated and physical drones to produce the types of evidence that would be needed for a certification argument that the timing requirements for safety have been met.

Evaluation with a drone flight simulator. There are obviously serious difficulties inherent in evaluating new avionics software by using it to fly an actual drone (injury to humans, damage to property, drone damage or loss, *etc.*). We therefore plan to first test and evaluate the software we produce with *AirSim* from Microsoft Research [82], a widely used environment simulator for AI research on autonomous vehicles. We discuss later our plans for evaluations using a physical drone.

AirSim. AirSim uses the Unreal Engine by Epic Games to produce photo-realistic scenes along with accurate physical models of drone dynamics. As shown in Fig. 9, AirSim can be used as a remote procedure call (RPC)-based client-server system with the simulated environment and drone model(s) running on the server. The server provides to the client images of the simulated environment from multiple cameras and readings from sensors (*e.g.*, IMU, GPS). The server can also provide ground-truth data needed for evaluations, such as image depth and disparity, along with collision-event reports. The client can use image and sensor data to provide perception, planning, and control for flying, and send control parameters to the drone model for changing velocity, destination, *etc.*

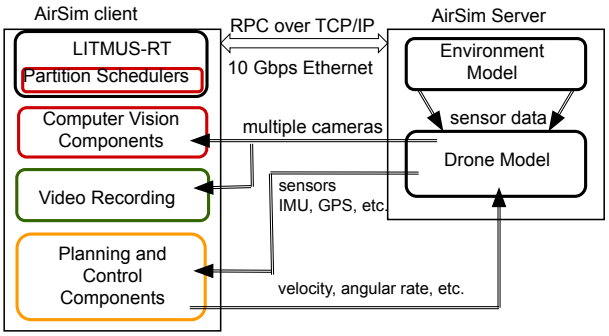

Figure 9: Diagram of AirSim components

We will configure our AirSim client as shown in Fig. 9 to use LITMUS^{RT} (see Sec. 3) augmented with the time and space isolation features we develop. These features will enable critical components on the client (*e.g.*, object detection) to run alongside, and be isolated from, less-critical components (*e.g.*, aerial video recording), without the excessive capacity loss of current ARINC 653 methods (see Fig. 2).

Evaluation methods. To evaluate our time and space partitioning methods, we will instantiate a subset of AI-based avionics components consisting of **(i)** CV for object recognition and tracking, and **(ii)** planning algorithms for navigating the drone to accomplish its task while avoiding obstacles. We will also include a non-AI component (video recording) to exercise the isolation from interfering non-critical workloads. We plan to use open-source implementations of these AI components adapted to run on our client platform.

AirSim's environment model provides several built-in environments of varying complexity but also allows for the creation of custom environments. AirSim was used for a drone racing competition called *Game of Drones* at NeurIPS 2019 [64]. Competitors could participate in challenges emphasizing perception, navigation, and collision avoidance. We plan to use their open source "race track" environments as a starting point for evaluations and then craft additional scenarios of varying complexity, which will include static and dynamic obstacles that the perception and planning algorithms will need to detect and consider. Our evaluation metrics will summarize the success of the drone in navigating its environment during a set interval of time without encountering unsafe events. The AirSim server reports unsafe events such as collisions or near-misses along with navigation errors. In addition to success metrics reported by AirSim, we will also monitor metrics relevant to our research, such as graph response times, cache and memory usage, *etc.* (*Monitoring* is actually an important part of FAA-approved certification.) In addition to the metrics used in the AirSim evaluations, we will use microbenchmarks throughout the research outlined in Sec. 3, notably to assess various provisioning choices and sharing vs. isolation tradeoffs. Success here will be demonstrated by showing that a given component's temporal safety is unaffected by other co-running components, even adversarial ones that stress any non-isolated hardware resources.

Evaluation using a sub-scale drone. Once our implementation has been sufficiently vetted in AirSim, we will conduct further evaluations using an actual drone. AirSim provides a portable library that is API compatible with their client-server API so software verified by simulation can move to a physical drone using a compatible flight controller (*e.g.*, Pixhawk [73]). We will use a commercial drone (*e.g.*, a kit compatible with Pixhawk [36]) and outfit it with sensors, battery, and an embedded computer system. Our drone will then be capable of operating autonomously, by a remote pilot, or as a combination of both.

Experiments to evaluate the drone's performance for autonomous object detection, tracking, and collision avoidance while following pre-planned paths will be conducted outdoors. We will conduct path-navigation trials in different lighting conditions and with a variety of obstacles. In some trials, the drone under test will share its airspace with a remote-piloted drone to test avoidance of moving objects. We will use most of the same metrics for success as in the AirSim simulations (*e.g.*, completion time and number of unsafe incidents on a given route), plus measurements such as response times.

Evaluation using Northrop Grumman facilities. PI Anderson has been collaborating with Northrop Grumman Aerospace Sector (NGAS) over the last two decades in the area of real-time systems schedulability and temporal assurance. This ongoing collaboration provides a sound basis for our mutual collaboration on this effort. NGAS's focus on airworthiness and certification of unmanned system avionics on multicore systems aligns well with the research goals of this effort. We expect to leverage NGAS's activities and facilities to evaluate and demonstrate concepts developed herein, including detailed aircraft simulations that can be used to drive and evaluate our prototypes. This work will be mostly done by our graduate students during summer internships at NGAS. We have had many interns work at NGAS in the past and will continue this relationship under this effort. As in the past, the challenge problems from NGAS that we work on will have IP-related details sanitized so that our results can be freely published.

5 Project Management and Collaboration Plan

This project brings together researchers in the areas of multicore and GPU computing (Anderson and Smith), operating systems (Anderson and Smith), real-time systems (Anderson), AI and Robotics (Alterovitz), and avionics (Sarathy). This collaboration plan explains how this diverse team will interact.

Roles. PI Anderson (UNC) will lead the overall effort, including coordination. His work on real-time systems and GPU management is central to bringing the project together. Co-PI Sarathy (NGAS) will provide guidance and supervision of identifying and shaping avionics challenge problems and supplying representative use cases. He will serve in a technical evaluator role, leveraging his years of experience at Northrop Grumman, to properly assess the potential utility of solutions developed in this effort. He will coordinate the demonstration of these capabilities, where applicable, to relevant NGAS programs for potential insertion and transition. This uniquely close connection to industry will be enhanced by students interning at Northrop Grumman under Sarathy's supervision. Co-PI Smith (UNC) will oversee the experimental work discussed in Sec. 4. His extensive background on GPU programming and OS development makes him ideally suited to lead this work. Co-PI Alterovitz (UNC) will oversee the AI refactoring work discussed in Sec. 3.4. He will bring to the project his considerable experience developing planning and control algorithms for autonomous systems.

Project management. The project will commence with an in-person kick-off meeting at UNC. Thereafter, bi-weekly Zoom meetings will be held. These meetings will be attended by all investigators and supported students and any other interested parties at Northrop Grumman. These meetings will be used to discuss progress, difficulties, and future plans. Additionally, subgroups will meet weekly to coordinate research with each of the investigators individually. Additional meetings will be scheduled on an ad hoc basis to produce papers. Some in-person meetings that involve travel may occasionally be held as well.

Internships. It is expected that one or more of the supported students will intern at Northrop Grumman each summer or longer (where practical). These students will evaluate the software and results produced in the project using facilities at Northrop Grumman, as explained in Sec. 4. They will also participate in safety-critical software projects to gain industry certification experience. A letter of commitment from Northrop Grumman is included with this submission.

Design repository. UNC will create and maintain a repository as a central place for information sharing. Artifacts generated, including design documents, technical reports, publications, use cases, presentations, test data, and software, will be stored and accessible for team members.

Dissertations and honors theses. This project is expected to form the basis for the dissertations of at least three Ph.D. students and the honors theses for several undergraduate students. (Undergraduate students do their thesis work as part of course work rather than paid positions.) Collaborations among the students will be encouraged by requiring them to work on joint papers for conference submission.

6 Broader Impacts

This project will have broader impacts on many fronts: it will provide needed "cross pollination" between academia and industry by having participants from both sectors; it will benefit students by providing

fuel for several new dissertations and undergraduate honors theses, and by providing industry internship opportunities; it will benefit teachers by providing new courseware; it will benefit the research community by contributing open-source software; it will benefit underrepresented groups through outreach efforts. The investigators have a sustained track record with respect to *all* of these activities.

We lack sufficient space to discuss each of these avenues for impact in detail. We have therefore opted to focus our attention here on three particular areas of impact that are somewhat unique to our project. (Additional details are provided in the supplemental document on *Broadening Participation in Computing*.)

Societal impact. The results of this project will provide a sound basis for informing the evolution of avionics certification standards as they embrace AI functionality. The PI has been involved in ongoing efforts by the Army, Navy, and Air Force to create such standards in the military sector, so more immediate impact can be expected there. The proposed research will also aid in educating the current engineering workforce with respect to the design and development of future control systems using advanced embedded hardware platforms. Such education will be done through meetings, workshops, and seminars with engineers on site at NGAS and at industry-focused events such as the AIAA Aviation Forum.

Impact within the research community and industry. We expect that the open-source component framework we intend to implement and maintain will transform how experimental research related to real-time avionics systems is done, much the way our development of LITMUS^{RT} transformed the way experimental work is done in the real-time systems community. We plan to host tutorial sessions to promote our open-source framework at top conferences such as the IEEE Real-Time Systems Symposium (RTSS). We also plan to host a workshop at an AI/robotics conference to discuss the importance and challenges of considering real-time constraints and certification in AI algorithms designed for autonomous drones. Additionally, Wind River is a key partner of Northrop Grumman, and this close working relationship offers an ideal avenue for exporting our research ideas, as implemented in LITMUS^{RT}, into a commercial RTOS.

Broadening the participation of women and girls in computing. To say that there is a dearth of women in real-time and embedded computing would be a vast understatement. When the PI became chair of the IEEE Technical Committee on Real-Time Systems (TCRTS) in 2016, he was asked to provide demographic statistics for the most recent flagship real-time conference, RTSS 2015. Of the 175 people who attended that event, only eight (yes, eight) were women! [6]

The UNC CS department features multiple groups and events that are already making efforts to broaden the participation of many underrepresented groups in computing. UNC's Graduate Women in CS (GWiCS) group hosts an annual research symposium to help undergraduate women and other under-represented minorities learn about research opportunities in CS. GWiCS also hosts Tar Heel Hack, in which local middle and high school girls participate in a six-hour hackathon. UNC CS also supports a Girls Who Code Club, which provides local girls in grades 6-12 with a community in which to learn about CS, and PearlHacks, which is one of the first all-female hackathons in the country. Additionally, UNC CS hosts an annual Open House and Science Expo, as well as Maze Day, which is an annual department event during which over 150 K-12 students with visual impairments visit to explore CS-related demos.

We intend to continue our involvement in these activities through the following actionable objectives:

- **Tar Heel Hack project and Maze Day experience.** We will incorporate some of our results into a Raspberry Pi and sensor-enabled Lego Robot hackathon project for future Tar Heel Hack events, to introduce the girls to various nuances of autonomy. We will also create an interactive robot building and sound-enabled motion debugging experience for Maze Day.
- **CS Department demos.** Using an F1/10-scale car and a home assistive robot, we have demoed our prior research for hundreds of local students at the UNC Science Expo and the CS Department Open House. In this vein, we will devise new demos involving autonomous drones, cars, and other robots for these events, as well as for Girls Who Code Club meetings, Tar Heel Hack, and PearlHacks.
- **New first-year undergraduate seminar course.** We will explore CPS topics ranging from control theory to path planning and AI applications in a new first-year undergraduate seminar course. We will also explore various aspects of the ethics of computing and other related topics.

[6] PI Anderson was the first TCRTS chair to create a Diversity Committee in an attempt to begin rectifying this problem.

- **Undergraduate and graduate CPS courses.** In recent years, we have expanded our course offerings in CPS at both the undergraduate and graduate levels. We will incorporate results from this project into these courses, and produce freely available teaching materials.
- **REU opportunities and local outreach.** We will provide summer REUs for students from local colleges and universities to contribute to our work, and reach out to these schools to give talks on research and the process of applying to graduate CS programs. (Between them, Anderson and Alterovitz have chaired the UNC CS Graduate Admissions Committee for the past 19 years.)

7 Results From Prior NSF Support

Co-PI Sarathy has not received prior NSF support.

Anderson and Smith. CPS 1837337, *CPS: Medium: GOALI: Real-Time Computer Vision in Autonomous Vehicles: Real Fast Isn't Good Enough*, $1,000,000, 10/1/2018-12/31/2021.

Intellectual merit. In this project (which will be in its final year when the proposed project starts), we have been investigating how to support real-time CV-based workloads on embedded automotive platforms. A key goal is to develop a real-time variant of OpenVX, a recently ratified CV API that allows algorithms to be specified in a graph-based way [52]. OpenVX has quickly emerged as the CV API of choice for embedded use cases, but lacks concepts relevant to real-time analysis, such as execution rates, priorities, *etc.* We have produced new research results on a number of topics, including techniques for increasing parallelism in CNN execution and GPU access [70, 100], scheduling methods and schedulability analysis for OpenVX [7, 8], the use of hardware threads to increase parallelism in real-time multicore systems [67–69], techniques for trading speed for accuracy in CNNs [95]. and algorithms and analysis for multiprocessor real-time scheduling and synchronization [2, 34, 65, 90]. In total, we have produced 12 papers under the auspices of this grant, including one best paper awardee [34].

Broader impacts. Some of the experimental work in this project has involved using an F1/10-scale autonomous car. This car has been demoed at the UNC Science Expo, the CS Department Open House, and various GWiCS events (see Sec. 6). Two of the students in this project were also members of a UNC team that won the F/10 Autonomous Racing Challenge at CPS Week 2019. The results of this project have been featured in invited talks given by the investigators at military venues, universities, and conferences. Three Ph.D. students have been supported by this project; all are expected to complete their degrees in the next year. Two undergraduate students have also conducted research related to this project. Three graduate students have worked as summer interns at General Motors Research as part of this project.

Alterovitz. CCF-1533844, *XPS: FULL: DSD: Parallel Motion Planning for Cloud-Connected Robots*, $670,536, 9/1/2015–8/31/2020.

Intellectual merit. We developed a new software framework for motion planning for cloud-connected robots that effectively parallelizes motion planning and distributes the computation across the robot's embedded processor and cloud-based multicore servers. Our approach enables battery-powered robots, which may not have the compute power to quickly solve complex motion-planning problems using their embedded processor, to leverage the cloud while accounting for network latency and bandwidth limitations [38, 40–42]. We also developed algorithms that parallelize robot motion planning when executing tasks learned from human-provided demonstrations [14, 15]. We also created motion planning templates (MPT), an open-source robot motion planning library to share many of the results of this project [39].

Broader impacts. Our new algorithms and software have the potential to enable robots to autonomously complete tasks in new domains where the challenge of motion planning is currently prohibitive due to power constraints on compact mobile robots, including home assistance for the elderly and people with disabilities. Results from the project were presented in two Ph.D. dissertations, an undergraduate honors thesis, and over 10 research workshops and events. We also presented live demonstrations of the results of this project at the UNC Science Expo and other outreach events attended by hundreds of school children and community members with the goal of inspiring interest in robotics and computer science.

References

[1] "LITMUS^{RT} home page," Online at http://www.litmus-rt.org/, 2020.

[2] S. Ahmed and J. Anderson, "A soft-real-time optimal semi-clustered scheduler with a constant tardiness bound," in *Proceedings of the 26th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, August (to appear) 2020.

[3] A. Alhammad and R. Pellizzoni, "Trading cores for memory bandwidth in real-time systems," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2016, pp. 1–11.

[4] A. Alhammad, S. Wasly, and R. Pellizzoni, "Memory efficient global scheduling of real-time tasks," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2015, pp. 285–296.

[5] S. Altmeyer, R. Douma, W. Lunniss, and R. Davis, "Evaluation of cache partitioning for hard real-time systems," in *Proceedings of the 26th Euromicro Conference on Real-Time Systems*, 2014, pp. 15–26.

[6] *AMD64 Technology Platform Quality of Service Extensions*, AMD Corporation, 2018, obtained online at https://developer.amd.com/wp-content/resources/56375_1.00.pdf.

[7] T. Amert, S. Voronov, and J. Anderson, "OpenVX and real-time certification: The troublesome history," in *Proceedings of the 40th IEEE Real-Time Systems Symposium*, December 2019, pp. 312–325.

[8] T. Amert, M. Yang, S. Nandi, T. Vu, J. Anderson, , and F. Smith, "The price of schedulability in multi-object tracking: The history-vs.-accuracy trade-off," in *Proceedings of the 23rd International Symposium on Real-Time Distributed Computing*, May 2020, pp. 124–133.

[9] J. Anderson, S. Baruah, and B. Brandenburg, "Multicore operating-system support for mixed criticality," in *Proceedings of the Workshop on Mixed Criticality: Roadmap to Evolving UAV Certification*, 2009.

[10] N. Audsley, "Memory architecture for NoC-based real-time mixed criticality systems," in *Proceedings of the First Workshop on Mixed Criticality*, 2013, pp. 37–42.

[11] A. A. Balkema and L. De Haan, "Residual life time at great age," *The Annals of Probability*, pp. 792–804, 1974.

[12] S. Baruah, "Federated scheduling of sporadic DAG task systems," in *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, 2015, pp. 179–186.

[13] S. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, L. Stougie, and A. Wiese, "A generalized parallel task model for recurrent real-time processes," in *2012 IEEE 33rd Real-Time Systems Symposium*. IEEE, 2012, pp. 63–72.

[14] C. Bowen and R. Alterovitz, "Closed-Loop Global Motion Planning for Reactive, Collision-Free Execution of Learned Tasks," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 7, no. 1, pp. 10:1—-10:16, may 2018. [Online]. Available: http://doi.acm.org/10.1145/3209045

[15] C. Bowen and R. Alterovitz, "Probability-weighted temporal registration for improving robot motion planning and control learned from demonstrations," in *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, dec 2018, pp. 1–16.

[16] F. J. Cazorla, L. Kosmidis, E. Mezzetti, C. Hernandez, J. Abella, and T. Vardanega, "Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 14:1–14:35, Feb. 2019.

[17] M. Chisholm, N. Kim, S. Tang, N. Otterness, J. Anderson, F. Smith, and D. Porter, "Supporting mode changes while providing hardware isolation in mixed-criticality multicore systems," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, 2017.

[18] M. Chisholm, N. Kim, B. Ward, N. Otterness, J. Anderson, and F. Smith, "Reconciling the tension between hardware isolation and data sharing in mixed-criticality, multicore systems," in *Proceedings of the 37th IEEE Real-Time Systems Symposium*, 2016.

[19] M. Chisholm, B. Ward, N. Kim, and J. Anderson, "Cache sharing and isolation tradeoffs in multicore mixed-criticality systems," in *Proceedings of the 36th IEEE Real-Time Systems Symposium*, 2015, pp. 305–316.

[20] H. Choset, K. M. Lynch, S. A. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[21] R. Davis and L. Cucu-Grosjean, "A survey of probabilistic timing analysis techniques for real-time systems," *Leibniz Transactions on Embedded Systems*, vol. 6, no. 1, pp. 03–1–03:60, 2019.

[22] Z. Dong and C. Liu, "An efficient utilization-based test for scheduling hard real-time sporadic DAG task systems on multiprocessors," in *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019, pp. 181–193.

[23] "Multi-core processors," https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/software, FAA, 2016.

[24] "Regulations and policies," https://www.faa.gov/regulations_policies/, FAA, 2020.

[25] "UAS by the numbers," https://www.faa.gov/uas/resources/by_the_numbers/, FAA, 2020.

[26] *System Safety Handbook*, Federal Aviation Administration (FAA), 2011.

[27] R. A. Fisher and L. H. C. Tippett, "Limiting forms of the frequency distribution of the largest or smallest member of a sample," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 24. Cambridge University Press, 1928, pp. 180–190.

[28] J. Forget, F. Boniol, E. Grolleau, D. Lesens, and C. Pagetti, "Scheduling dependent periodic tasks without synchronization mechanisms," in *Proceedings of the 16th Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2010, pp. 301–310.

[29] G. Giannopoulou, N. Stoimenov, P. Huang, and L.Thiele, "Scheduling of mixed-criticality applications on resource-sharing multicore systems," in *Proceedings of International Conference on Embedded Software*, 2013, pp. 17:1–17:15.

[30] A. Giusti, J. Guzzi, D. C. Ciresan, F. L. He, J. P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[31] M. Hassan and H. Patel, "Criticality- and requirement-aware bus arbitration for multi-core mixed criticality systems," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2016, pp. 1–11.

[32] M. Hassan, H. Patel, and R. Pellizzoni, "A framework for scheduling DRAM memory accesses for multi-core mixed-time critical systems," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2015, pp. 307–316.

[33] J. Herter, P. Backes, F. Haupenthal, and J. Reineke, "CAMA: A predictable cache-aware memory allocator," in *Proceedings of the 23rd Euromicro Conference on Real-Time Systems*, 2011, pp. 23–32.

[34] C. Hobbs, Z. Tong, and J. Anderson, "Concurrency groups: A new way to look at real-time multiprocessor lock nesting," in *Proceedings of the 27th International Conference on Real-Time Networks and Systems*, September 2019, pp. 112–122.

[35] P. Holman and J. Anderson, "Locking under Pfair scheduling," *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 140–174, 2006.

[36] Holybro, "Holybro X500 kit compatible with Pixhawk 4." [Online]. Available: https://shop.holybro.com/x500-kit_p1180.html

[37] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[38] J. Ichnowski and R. Alterovitz, "Concurrent Nearest-Neighbor Searching for Parallel Sampling-based Motion Planning in SO(3), SE(3), and Euclidean Spaces," in *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018.

[39] J. Ichnowski and R. Alterovitz, "Multilevel Incremental Roadmap Spanners for Reactive Motion Planning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2019, pp. 1504–1509.

[40] J. Ichnowski, W. Lee, V. Murta, S. Paradis, R. Alterovitz, J. E. Gonzalez, I. Stoica, and K. Goldberg, "Fog robotics algorithms for distributed motion planning using lambda serverless computing," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020.

[41] J. Ichnowski, J. Prins, and R. Alterovitz, "Cloud-based motion plan computation for power-constrained robots," in *Algorithmic Foundations of Robotics (WAFR 2016)*, dec 2016.

[42] J. Ichnowski, J. Prins, and R. Alterovitz, "The Economic Case for Cloud-based Computation for Robot Motion Planning," in *Proc. International Symposium on Robotics Research (ISRR)*, dec 2017.

[43] D. Iorga, T. Sorensen, J. Wickerson, and A. F. Donaldson, "Slow and steady: Measuring and tuning multicore interference," in *Proceedings of the 26th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2020.

[44] S. Jain, I. Baek, S. Wang, and R. Rajkumar, "Fractional GPUs: Software-based compute and memory bandwidth reservation for GPUs," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2019, pp. 29–41.

[45] J. Jalle, E. Quinones, J. Abella, L. Fossati, M. Zulianello, and P. Cazorla, "A dual-criticality memory controller (DCmc) proposal and evaluation of a space case study," in *Proceedings of the 35th IEEE Real-Time Systems Symposium*, 2014, pp. 207–217.

[46] X. Jiang, N. Guan, D. Liu, and W. Liu, "Analyzing GEDF scheduling for parallel real-time tasks with arbitrary deadlines," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1537–1542.

[47] X. Jiang, N. Guan, X. Long, and W. Yi, "Semi-federated scheduling of parallel real-time tasks on multiprocessors," in *Proceedings of the 38th IEEE Real-Time Systems Symposium*, 2017, pp. 80–91.

[48] X. Jiang, J. Sun, Y. Tang, and N. Guan, "Utilization-tensity bound for real-time DAG tasks under global EDF scheduling," *IEEE Transactions on Computers*, vol. 69, no. 1, pp. 39–50, 2019.

[49] S. Jiménez Gil, I. Bate, G. Lima, L. Santinelli, A. Gogonel, and L. Cucu-Grosjean, "Open challenges for probabilistic measurement-based worst-case execution time," *IEEE Embedded Systems Letters*, vol. 9, no. 3, pp. 69–72, Sep. 2017.

[50] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 66–75.

[51] R. E. Kessler and M. D. Hill, "Page placement algorithms for large real-indexed caches," *ACM Transactions on Computer Systems (TOCS)*, vol. 10, no. 4, pp. 338–359, 1992.

[52] Khronos Group, "OpenVX homepage," Online at https://www.khronos.org/openvx/, 2018.

[53] H. Kim, D. Broman, E. Lee, M. Zimmer, A. Shrivastava, and J. Oh, "A predictable and command-level priority-based DRAM controller for mixed-criticality systems," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2015, pp. 317–326.

[54] H. Kim, D. de Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar, "Bounding memory interference delay in COTS-based multi-core systems," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2014, pp. 145–154.

[55] H. Kim, A. Kandhalu, and R. Rajkumar, "A coordinated approach for practical OS-level cache management in multi-core real-time systems," in *Proceedings of the 25th Euromicro Conference on Real-Time Systems*, 2013, pp. 80–89.

[56] N. Kim, M. Chisholm, N. Otterness, J. Anderson, and F. Smith, "Allowing shared libraries while supporting hardware isolation in multicore real-time systems," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2017.

[57] N. Kim, S. Tang, N. Otterness, J. Anderson, F. D. Smith, and D. Porter, "Supporting I/O and IPC via fine-grained OS isolation for mixed-criticality real-time tasks," in *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, 2018.

[58] N. Kim, B. Ward, M. Chisholm, C.-Y. Fu, J. Anderson, and F. Smith, "Attacking the one-out-of-*m* multicore problem by combining hardware management with mixed-criticality provisioning," in *Real-Time Embedded Technology and Applications Symposium*. IEEE, 2016, pp. 49–160.

[59] N. Kim, B. Ward, M. Chisholm, C.-Y. Fu, J. Anderson, and F. Smith, "Attacking the one-out-of-m multicore problem by combining hardware management with mixed-criticality provisioning," *Real-Time Systems*, vol. 53, pp. 709–759, 2017.

[60] O. Kotaba, J. Nowotsch, M. Paulitsch, S. Petters, and H. Theiling, "Multicore in real-time systems – temporal isolation challenges due to shared resources," in *Proceedings of the Conference on Design, Automation and Test in Europe, WICERT*, 2013.

[61] Y. Krishnapillai, Z. Wu, and R. Pellizzoni, "ROC: A rank-switching, open-row DRAM controller for time-predictable systems," in *Proceedings of the 26th Euromicro Conference on Real-Time Systems*, 2014, pp. 27–38.

[62] J. Li, A. Saifullah, K. Agrawal, and C. Gill, "Capacity augmentation bound of federated scheduling for parallel DAG tasks," Department of Computer Science and Engineering, Washington University in St Louis, Tech. Rep. WUCSE-2014-44, 2014.

[63] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to Fly by Driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[64] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "AirSim drone racing lab," 2020.

[65] C. Nemitz, T. Amert, M. Goyal, and J. Anderson, "Concurrency groups: A new way to look at real-time multiprocessor lock nesting," in *Proceedings of the 27th International Conference on Real-Time Networks and Systems*, September 2019, pp. 187–197.

[66] B. Ocker, "FAA special topics," in *Collaborative Workshop: Solutions for Certification of Multicore Processors*, Nov. 2018.

[67] S. Osborne, S. Ahmed, S. Nandi, and J. Anderson, "Exploiting simultaneous multithreading in priority-driven hard real-time systems," in *Proceedings of the 26th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, August (to appear) 2020.

[68] S. Osborne and J. Anderson, "Simultaneous multithreading and hard real time: Can it be safe?" in *Proceedings of the 32nd Euromicro Conference on Real-Time Systems*, July (to appear) 2020.

[69] S. Osborne, J. Bakita, and J. Anderson, "Simultaneous multithreading applied to real time," in *Proceedings of the 31st Euromicro Conference on Real-Time Systems*, July 2019, pp. 3:1–3:22.

[70] N. Otterness and J. Anderson, "AMD GPUs as an alternative to nvidia for supporting real-time workloads," in *Proceedings of the 32nd Euromicro Conference on Real-Time Systems*, July (to appear) 2020.

[71] R. Pellizzoni, A. Schranzhofer, J. Chen, M. Caccamo, and L. Thiele, "Worst case delay analysis for memory interference in multicore systems," in *Proceedings of Design, Automoation, and Test in Europe*, 2010, pp. 741–746.

[72] J. Pickands III *et al.*, "Statistical inference using extreme order statistics," *the Annals of Statistics*, vol. 3, no. 1, pp. 119–131, 1975.

[73] "Pixhawk: The open standards for drone hardware," https://pixhawk.org/, Pixhawk, 2020.

[74] P. J. Prisaznuk, "Integrated modular avionics," in *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference@ m_NAECON 1992*. IEEE, 1992, pp. 39–45.

[75] P. J. Prisaznuk, "ARINC 653 role in integrated modular avionics (IMA)," in *2008 IEEE/AIAA 27th Digital Avionics Systems Conference*. IEEE, 2008, pp. 1–E.

[76] RTCSA-SC, "205/eurocae wg-71. Software considerations in airborne systems and equipment certification," *No. RTCA DO-178C, RTCA Inc*, vol. 1140, 2011.

[77] F. Sadeghi and S. Levine, "CADˆ2 RL : Real Single-Image Flight Without a Single Real Image," in *Proc. Robotics: Science and Systems*, jul 2017.

[78] SAE-Aerospace, "Guidelines for development of civil aircraft and systems," *Bericht ARP4754, SAE International*, 2010.

[79] Y. Saito, F. Sato, T. Azumi, S. Kato, and N. Nishio, "ROSCH: Real-time scheduling framework for ROS," in *Proceedings of the 22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2018, pp. 52–58.

[80] J. G. Saw, M. C. Yang, and T. C. Mo, "Chebyshev inequality with estimated mean and variance," *The American Statistician*, vol. 38, no. 2, pp. 130–132, 1984.

[81] M. A. Serrano, A. Melani, M. Bertogna, and E. Quiñones, "Response-time analysis of DAG tasks under fixed priority scheduling with limited preemptions," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 1066–1071.

[82] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.0506

[83] "Autopilot description," https://www.skybrary.aero/index.php/AutopilotDescription, Skybrary, 2020.

[84] "Skydio 2," https://www.skydio.com/, Skydio, 2020.

[85] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward low-flying autonomous {MAV} trail navigation using deep neural networks for environmental awareness," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, pp. 4241–4247, 2017.

[86] "Arp-4761–guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment," Society of Automotive Engineers (SAE), 1996.

[87] W. Sun, J. {van den Berg}, and R. Alterovitz, "Stochastic extended {LQR} for optimization-based motion planning under uncertainty," *IEEE Trans. Automation Science and Engineering*, vol. 13, no. 2, pp. 437–447, apr 2016.

[88] Y. Suzuki, T. Azumi, S. Kato *et al.*, "HLBS: Heterogeneous laxity-based scheduling algorithm for DAG-based real-time computing," in *Proceedings of the 4th International Conference on Cyber-Physical Systems, Networks, and Applications*. IEEE, 2016, pp. 83–88.

[89] R. Tabish, R. Mancuso, S. Wasly, A. Alhammad, S. Phatak, R. Pellizzoni, and M. Caccamo, "A real-time scratchpad-centric OS for multi-core embedded systems," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2016, pp. 1–11.

[90] S. Tang, S. Voronov, and J. Anderson, "GEDF tardiness: Open problems involving uniform multiprocessors and affinity masks resolved," in *Proceedings of the 31st Euromicro Conference on Real-Time Systems*, July 2019, pp. 13:1–13:21.

[91] P. Traverse, I. Lacaze, and J. Souyris, "Airbus fly-by-wire: A process toward total dependability," in *ICAS 2006, 25th International Congress of the Aeronautical Sciences*, 2006.

[92] T. Trepetch, "Arm Neoverse E1 Platform: Empowering the infrastructure to meet next generation throughput demands," Feb 2019.

[93] M. Verucchi, M. Theile, M. Caccamo, and M. Bertogna, "Latency-aware generation of single-rate DAGs from multi-rate task sets," in *Proceedings of the 26th IEEE Real Time and Embedded Technology and Applications Symposium*, 2020, pp. 107–118.

[94] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proceedings of the 28th IEEE Real-Time Systems Symposium*, 2007, pp. 239–243.

[95] T. Vu, M. Eder, T. Price, and J.-M. Frahm, "Any-width networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[96] B. Ward, J. Herman, C. Kenna, and J. Anderson, "Making shared caches more predictable on multicore platforms," in *Proceedings of the 25th Euromicro Conference on Real-Time Systems*, 2013, pp. 157–167.

[97] M. Xu, L. T. X. Phan, H.-Y. Choi, and I. Lee, "Analysis and implementation of global preemptive fixed-priority scheduling with dynamic cache allocation," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2016.

[98] K. Yang, G. Elliott, and J. Anderson, "Analysis for supporting real-time computer vision workloads using OpenVX on multicore+GPU platforms," in *Proceedings of the 23rd International Conference on Real-Time Networks and Systems*, 2015, pp. 77–86.

[99] M. Yang, T. Amert, K. Yang, N. Otterness, J. Anderson, F. D. Smith, and S. Wang, "Making OpenVX really 'real time'," in *Proceedings of the 39th IEEE Real-Time Systems Symposium*, 2018.

[100] M. Yang, S. Wang, J. Bakita, T. Vu, F. Smith, J. Anderson, and J.-M. Frahm, "Re-thinking cnn frameworks for time-sensitive autonomous-driving applications: Addressing an industrial challenge," in *Proceedings of the 25th IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2019, pp. 305–317.

[101] H. Yun, R. Mancuso, Z. Wu, and R. Pellizzoni, "PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms," in *Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2014, pp. 155–166.

[102] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "Memory access control in multiprocessor for real-time systems with mixed criticality," in *Proceedings of the 24th Euromicro Conference on Real-Time Systems*, 2012, pp. 299–308.

Facilities, Equipment, and Other Resources

UNC Facilities

General computing environment. The department's computing environment includes over 1,000 computers, ranging from older systems used for generating network traffic for simulated Internet experiments to state-of-the-art workstations and clusters for graphics and compute-intensive research. Departmental servers provide computing services, disk space, email, source code repositories, web services, database services, backups, in addition to other services. All systems are integrated by means of high-speed networks, described below, and are supported by highly skilled technical staff who provide a consistent computing environment throughout the department. Most students are assigned to a two- or three-person office, though we also have larger offices that can hold more students. Each student is assigned a computer, with computer assignments based on the students' research or teaching assignments and their seniority. Currently the minimum configuration for student computers is a desktop system with dual-core 64-bit 2-GHz processor, 4-GB RAM, and 240-GB hard drive, though many students have computers that exceed these specifications depending on their job and seniority. In addition to the departmental servers and office systems, our research laboratories contain a wide variety of specialized equipment and facilities. We also have an extensive virtual machine infrastructure spanning two machine rooms.

General computing systems include 800+ Intel-based computers plus about 50 Macintosh systems, plus a variety of servers and virtual machines.

Software environment. Our systems primarily run the Windows 10 operating system, with some still running Windows 7. We have a large number of systems, including many of the servers, running Ubuntu or Red Hat Linux, and MacOS. We use the AFS file system for central file storage. Languages most commonly used include J++, C++, Java, Python, PHP, and C. Document preparation is usually accomplished with standard applications on PC systems. Our extensive software holdings are continually evolving. We are a Google shop and have unlimited storage in Google Drive and use all core Google applications. UNC also offers Office365 cloud services.

Network environment. The network infrastructure available to the Computer Science Department is extensive. In the Frederick P. Brooks Jr. Building and Sitterson Hall, every office and common space is equipped with category 6 or better twisted pair cable. All offices in Sitterson also have coaxial and fiber optic cable. All data connections in both buildings supports 1 Gigabit per second, and we have a growing number of systems using 10 Gigabit connections. Extensive riser connections enable the department to create multiple separate physical networks between any points in the two buildings. Wifi networks over 802.11 A and N are available throughout both buildings.

A 10 Gigabit link connects the department's network to the UNC-CH campus network and the North Carolina Research and Education Network (NC-REN), allowing users to access the National LambdaRail/ Internet 2 network and commodity Internet. We also have a 10 gigabit connection through the Renaissance Computing Institute (RENCI) to the Global Environment for Network Innovations (GENI) project. The department's network is connected to the North Carolina Research and Education Network (NC-REN), a statewide network that links research and educational institutions. Our two-way video classroom and teleconference rooms allow connection to any institution served by the network.

UNC-provided computation infrastructure. The University of North Carolina at Chapel Hill provides computing resources for researchers at UNC-Chapel Hill. Among these are:

- The Longleaf cluster is a Linux-based computing system available to researchers across the campus free of charge. With nearly 6,500 conventional compute cores delivering 13,000 threads (hyperthreading is enabled) and a large, fast scratch disk space, it provides an environment that is optimized for memory and I/O intensive, loosely coupled workloads with an emphasis on aggregate job throughput over individual job performance. In particular, workloads consisting of a large quantity of jobs

each requiring a single compute host are best suited to Longleaf. The Longleaf cluster is targeted for data science and statistical computing workloads, very large memory jobs, and GPU computing. Resources are managed through a fair-share algorithm using SLURM as the resource manager/job scheduler.

- The Dogwood cluster is a Linux-based computing system available to researchers across the campus free of charge. With over 11,000 compute cores, a low latency, high bandwidth Infiniband EDR switching fabric and a large scratch disk space, it provides an environment that is optimized for large, multi-node, tightly coupled computational models typically using a message passing (*e.g.*, MPI) or hybrid (OpenMP + MPI) programming model. Most of the cluster is comprised of nodes with Intel Xeon processors with the Broadwell-EP micro-architecture and 44 cores per node. There is a partition with some newer Intel Skylake nodes as well as a smaller knl partition, with Intel Xeon Phi processors (Knight's Landing) for development purposes. Resources are managed through a fair-share algorithm using SLURM as the resource manager/job scheduler.
- Carolina Cloud Apps, a web application development platform powered by RedHat's OpenShift Enterprise software.
- The Virtual Computing Lab (VCL), which was originally developed by N.C. State University in collaboration with IBM. The VCL provides users with anytime, anywhere access to custom application environments created specifically for their use. VCL provides users remote access to hardware and software that they would otherwise have to install themselves on their own systems, or visit a computer lab to use. It also reduces the burden on computer labs to maintain large numbers of applications on individual lab computers, where in many cases it is difficult for some applications to coexist on the same machine. In the VCL, operating system images with the desired applications and custom configurations are stored in an image library, and deployed to a server on-demand when a user requests it.
- Participation in the RHEDcloud Foundation, which stands for Research, Healthcare and Higher Education Cloud Foundation. The RHEDcloud Project brings people together from research universities, pharma, healthcare providers, cloud service providers, and cloud platform providers to design and implement better security, automation, and integration for cloud computing.
- The Secure Research Workspace, which contains computational and storage resources specifically designed for management and interaction with high-risk data. The SRW is used for storage and access to Electronic Health Records (EHR) and other highly sensitive or regulated data; it includes technical and administrative controls that satisfy applicable institutional policies. SRW is specifically designed to be an enclave that minimizes the risk of storing and computing on regulated or sensitive data.

Real-time computing laboratory equipment. The Real-Time Systems Group has six general-purpose x86 machines, with a mix of servers and workstation systems. These machines feature CPUs from both Intel and AMD, with core counts ranging from four to 26. Additionally, the group has six recent ARM-based "Jetson" embedded platforms produced by NVIDIA, in addition to several less-powerful embedded ARM platforms from NVIDIA and other manufacturers. All of our workstation and server systems are equipped with GPUs, including the current top-of-the-line consumer GPUs from both NVIDIA and AMD.

We routinely use the research clusters listed above to run large-scale, overhead-aware schedulability studies. These studies, which can take many days to complete, involve assessing schedulability trends by examining many millions of randomly generated task systems with measured overheads affecting schedulability considered. These overhead measurements are taken on the lab machines noted above.

Robotics laboratory The UNC Robotics Laboratory is an over 800 square foot space dedicated to research, teaching, and outreach related to robots and autonomous systems. The laboratory includes a

Rethink Robotics Baxter Robot (a robot with two compliant arms, enabling it to operate safely in environments with humans), a Fetch mobile manipulator robot (which has a mobile base, a robotic arm, a lift joint, and a built-in RGB-D sensor), a SoftBank Robotics Nao robot (a child-size humanoid robot with articulated legs and arms), prototype medical robots, numerous iRobot Roomba platforms for multi-robot coordination experiments, a differential drive Pioneer P3-DX mobile robot from Mobile Robots, Inc., ceiling mounted cameras for robot tracking, Microsoft Kinects for RGB-D sensing, and an open space for indoor experiments on robots and autonomous systems. The laboratory also includes 3D graphics workstations featuring state-of-the-art multicore processors and GPUs as well as Phantom haptic devices for research on motion planning, control, and simulation of robots and autonomous systems. In addition to research, the laboratory is also used for teaching and outreach.

Northrop Grumman Facilities

Northrop Grumman Aerospace Sector (NGAS) is a premier provider of military aircraft, autonomous systems, aerospace structures, and next-generation surveillance, strike, and commercial solutions. From sea, air, land, or space, NGAS provides solutions that advance technology and discovery while meeting customer needs with high impact, best-value aeronautics products, systems, and services. Working closely with customers and suppliers, these systems technologies are used for a wide range of missions including intelligence, surveillance, and reconnaissance; protected communications; battle management; strike operations; electronic warfare; and missile defense. NGAS has industry-leading facilities for the design and testing of autonomous aerial vehicles, including simulators to evaluate vehicle physical components, control algorithms, various electronic devices, and mission flight scenarios. Co-PI Sarathy from NGAS will directly support the research initiatives outlined in this proposal (up to 5% FTE), collaborating with the research team to provide real-world scenarios, system requirements, and guidance on solution development and experimental platforms.

Data Management Plan

This data management plan covers all of the data products to be produced in this project. It includes **(i)** the source code for the LITMUS^{RT}-based component scheduling framework we propose to implement along with the associated tools for analyzing schedulability, **(ii)** source code that implements the avionics algorithms we propose to design, **(iii)** timing-analysis tools supporting execution-time measurements, **(iv)** the suite of test programs and scripts used to test and evaluate the software we produce, and **(v)** the data collected during evaluation experiments.

Source Code

As has been our practice, the source code we produce, along with supporting libraries, scripts, *etc.*, will be published under an open source GPL version 2 license (the conventional open-source license for Linux distribution) and will be freely available for direct download from public web servers maintained by the UNC Computer Science Department.

Additional test programs, scripts, and tools will be freely available under an open-source license (to be determined, but probably the BSD license) and directly downloadable from public web servers maintained by the UNC Computer Science Department. The intent in making these evaluation materials freely available is that other projects can independently verify evaluation results. While all these artifacts may be useful to other researchers, we emphasize that they are not intended to constitute a testbed and we do not provide support for testbed use.

Evaluation Data

Raw data from evaluation experiments can be several terabytes in size. It is thus currently infeasible to make this data available to the public through direct download from UNC. However, such evaluation data will be stored and backed up on secured servers maintained by the UNC Computer Science Department. Arrangements can be made with interested parties who may request copies of this data. These requests may be serviced through private direct download or through physical media delivered through the USPS or other carrier.

In conference and journal papers published by our research group in the past, distilled information from raw evaluation data has be made available through summary graphs, sometimes numbering in the thousands, in online appendices. We will continue this practice. These publications with extended appendices will be freely available for direct download from public web servers maintained by the UNC Computer Science Department.

Privacy

No personal data will be captured in this project. There are no privacy concerns for the storage of project data.

Long-Term Storage

All source code and data products described above will also be archived in the UNC Information Technology Services research computing repository for permanent storage.

UNC/Northrop Grumman Interactions

The proposed project will not generate proprietary data. Any data from Northrop Grumman shared with the research team will be sanitized without loss of its representativeness. Any experimental results obtained using Northrop Grumman facilities will be disseminated through publications and reports that are similarly sanitized and do not reveal confidential information. PI Anderson's research group at UNC has worked with co-PI Sarathy's team at Northrop Grumman repeatedly over the last two decades on a variety of projects. If the proposed effort is funded, we will put a new IP agreement between UNC and Northrop Grumman in place that is in line with agreements we have operated under in the past.

Broadening Participation in Computing Plan

Context. Women are critically underrepresented in computer science. This underrepresentation begins long before research conferences. At the undergraduate level, 25% of computer science students at UNC are female. This stands in contrast to the overall undergraduate student population, of which 59.4% are female (`https://oira.unc.edu/reports/`). This disparity motivates us to work toward increasing female participation in computing at the undergraduate level.

Target. This BPC plan focuses on efforts to increase female participation in computing at the undergraduate level, and to provide awareness and guidance for female undergraduate students concerning research opportunities. Emphasis will be placed on *critical thinking* and *writing skills*. Why an emphasis on writing? As Dr. Leslie Lamport aptly put it, "If you can't write, it won't compute" (`https://www.brandeis.edu/now/2017/may/commencement-lamport.html`).

Strategy. When the PI was a Ph.D. student at the University of Texas, Prof. Edsger Dijkstra led a weekly meeting called the *Tuesday Afternoon Club*. At each of these meetings, a single paper was read—*out loud*—and discussed. There were two basic assumptions underlying these meetings. First, no pre-meeting preparation was expected (unless the club was reading something you had written). Second, each considered paper was read and discussed only within the time limit of a single meeting—an unfinished paper was not considered further (of course, individual members could continue reading on their own).

These meetings were highly effective for at least three reasons. First, no preparation outside of meetings was required, so continued participation involved a *limited and predictable time commitment*. Second, listening to written text (particularly your own) being read and dissected out loud in real time is a highly effective exercise for honing one's writing skills. Third, by affording each paper strictly limited time, a broad exposure can be provided to a range of issues and topics in computing.

We propose to create and lead a UNC version of the Tuesday Afternoon Club that targets undergraduate female students. As all of the PIs in this project are male, we will also involve our female graduate students in this endeavor to provide the younger women with older female role models. We envision recruiting two or three first-year undergraduate women as new club members each year, with the hope that they will remain in the club until they graduate. Thus, once we reach steady state, the club would be comprised of eight to twelve undergraduate students. In addition to club meetings, each club member will be assigned one UNC PI as a mentor (with the mentoring load being evenly distributed). Each PI will meet with their mentees regularly (probably monthly).

A few details concerning this activity will require some additional thought. For example, freshman will not have the same level of knowledge as seniors. To account for this, we might need to interleave less advanced and more advanced material, with the younger students only attending meetings where material they can handle will be read. (In one of Dijkstra's meetings, geometry-related proofs one might encounter in high school were discussed.) Additionally, we will need to consider whether we should seed the club with some older students at the beginning instead of converging to a steady state.

Resources. At UNC, incoming freshmen take a first-year seminar (FYS). We intend to leverage these FYS offerings as a resource for recruiting club members. In particular, we will arrange to visit FYS classes and discuss this opportunity with students with the hope of attracting interest. Typically, students taking an FYS in the Computer Science Department will not have declared a major, so FYSs provide an opportunity to reach those students not already in a computer science class. Additionally, in recent FYS offerings in the Computer Science Department, 50% of the participants have been female.

Measurement and dissemination. As club members mature in their knowledge of computing, we intend to engage them in research activities of their own. It is our hope that each club member will produce an honors thesis (which will of course be read and discussed during club meetings) during their senior year. In accordance with the existing honors-thesis process, the results of each thesis will be disseminated via a poster at a public poster session and via a talk at a public honors colloquium event. (Hopefully, the impact of the club will be acknowledged during these events.)

A number of metrics can be tracked to demonstrate the viability of this BPC endeavor. These include: the number of students recruited each year, the length of time each recruited student remains as a club member, the number of members who become computer science majors, the number of members who produce an honors thesis, and the number of members who apply to graduate school. Regarding the latter, we intend to provide advice and guidance for applying to graduate school. (Between them, PI Anderson and co-PI Alterovitz have served as Director of Graduate Admissions in the Computer Science Department for the last 19 years, so they are very knowledgeable about the application process.) We also intend to provide guidance concerning summer internship opportunities.

A recent Data Buddies survey (`https://cra.org/cerp/data-buddies/`) revealed that female students felt less confident than their male peers in their ability to become a leader in the computing field or even to complete their undergraduate degree in computing—these differences were statistically significant. It is our belief that, by providing a safe space in our meetings for delving critically into both the technical content of the papers we read as well as how this content is presented, we can produce club members who are not lacking in confidence.